

1. Tempo disponibile 120 minuti.
2. Non è possibile consultare appunti, slide, libri, persone, siti web, ecc.
3. Scrivere in modo leggibile, su ogni foglio, nome, cognome e numero di matricola.
4. Le soluzioni agli esercizi che richiedono di progettare un algoritmo devono:
 - spiegare a parole l'algoritmo (se utile, anche con l'aiuto di esempi o disegni),
 - fornire e commentare lo pseudo-codice (indicando il significato delle variabili),
 - calcolare la complessità (con tutti i passaggi matematici necessari),
 - se l'esercizio ammette più soluzioni, a soluzioni computazionalmente più efficienti e/o concettualmente più semplici sono assegnati punteggi maggiori.

IMPORTANTE: Risolvere gli esercizi 1-2 e gli esercizi 3-4 su fogli separati. Infatti, al termine, dovrete consegnare gli esercizi 1-2 separatamente dagli esercizi 3-4.

1. Calcolare la complessità $T(n)$ del seguente algoritmo MYSTERY1:

Algorithm 1: MYSTERY1(INT n) \rightarrow INT

```

 $x = 0$ 
 $res = 1$ 
while  $n \geq 1$  do
   $n = n/3$ 
   $x = x + 1$ 
   $res = res \times 2 \times \text{MYSTERY2}(2x)$ 
return  $res$ 

function MYSTERY2(INT  $n$ )  $\rightarrow$  INT
if  $n \leq 1$  then
  return 1
else
   $x = 1$ 
  for  $i = 1, \dots, n$  do
     $x = 2 \times x$ 
  return  $\text{MYSTERY2}(2n/3) + x$ 

```

2. Consideriamo un array A contenente n interi compresi nell'intervallo $[1, k]$
 - a) Indicare il costo nel caso pessimo per ordinare A con
 - InsertionSort
 - MergeSort
 - CountingSort
 - b) Indicare qual è tra gli algoritmi di ordinamento al punto a) quello maggiormente efficiente (in termini di costo pessimo) assumendo di sapere che
 - $k = \Theta(n)$
 - $k = \Theta(n \log n)$
 - $k = \Theta(n^2)$
3. Su una base spaziale si trovano n astronauti che devono fuggire dalla base per un inconveniente tecnico. Per fuggire dalla base, sono disponibili k diverse navicelle di salvataggio. Ogni navicella

di salvataggio deve contenere esattamente un predeterminato numero di astronauti; in altri termini, data la navicella i -esima, con $i \in \{1, \dots, k\}$, tale navicella potrà contenere esattamente $N[i]$ astronauti. Progettare un algoritmo che dato il numero di astronauti n e l'array $N[1..k]$ che indica il numero di astronauti previsto su ognuna delle k navicelle, restituisce un booleano: *true* se esiste una combinazione di navicelle che permette di far fuggire esattamente tutti gli n astronauti, *false* altrimenti.

4. Dato un grafo orientato $G = (V, E)$ ed un vertice $v \in V$ bisogna calcolare il numero di vertici fortemente connessi con v . In altri termini, progettare un algoritmo che dati in input il grafo orientato $G = (V, E)$ e un vertice $v \in V$ restituisce in output la cardinalità del seguente insieme: $\{w \in V \mid w \text{ raggiungibile da } v \text{ e } v \text{ raggiungibile da } w\}$.